

CLAIMS

What is claimed is:

- 1 1. A transform system for graphics processing, comprising:
 - 2 (a) an input buffer adapted for being coupled to a vertex attribute buffer for
 - 3 receiving vertex data therefrom;
 - 4 (b) a multiplication logic unit having a first input coupled to an output of the
 - 5 input buffer;
 - 6 (c) an arithmetic logic unit having a first input coupled to an output of the
 - 7 multiplication logic unit;
 - 8 (d) a register unit having an input coupled to an output of the arithmetic logic
 - 9 unit;
 - 10 (e) an inverse logic unit including an input coupled to the output of the
 - 11 arithmetic logic unit or the register unit for performing an inverse or an
 - 12 inverse square root operation;
 - 13 (f) a conversion module coupled between an output of the inverse logic unit and
 - 14 a second input of the multiplication logic unit, the conversion module
 - 15 adapted to convert scalar vertex data to vector vertex data;
 - 16 (g) memory coupled to the multiplication logic unit and the arithmetic logic unit,
 - 17 the memory having stored therein a plurality of constants and variables for
 - 18 being used when processing the vertex data; and
 - 19 (h) an output converter coupled to the output of the arithmetic logic unit and
 - 20 adapted for being coupled to a lighting module to output the processed vertex
 - 21 data thereto.
- 1 2. The system as recited in claim 1, wherein the memory is coupled to the
- 2 second input of the multiplication logic unit.
- 1 3. The system as recited in claim 1, wherein the input of the inverse logic unit is
- 2 coupled to the output of the arithmetic logic unit.

1 4. The system as recited in claim 1, wherein the inputs of the multiplication
2 logic unit include multiplexers.

1 5. The system as recited in claim 1, wherein at least one of the inputs of the
2 arithmetic logic unit includes a multiplexer.

1 6. The system as recited in claim 1, wherein the memory has a write terminal
2 coupled to the output of the arithmetic logic unit.

1 7. The system as recited in claim 1, wherein the output of the multiplication
2 logic unit has a feedback loop coupled to the first input thereof.

1 8. The system as recited in claim 1, wherein the output of the register unit is
2 coupled to the first input of the multiplication logic unit.

1 9. The system as recited in claim 8, wherein the output of the register unit is
2 coupled to the second input of the multiplication logic unit.

1 10. The system as recited in claim 1, wherein the output of the arithmetic logic
2 unit has a feedback loop connected to the second input thereof.

1 11. The system as recited in claim 10, wherein the feedback loop has a delay
2 coupled thereto.

1 12. The system as recited in claim 1, wherein the multiplication logic unit is
2 capable of performing a rotate operation on vector vertex data.

1 13. The system as recited in claim 1, wherein the inverse logic unit is capable of
2 clamping a value of an inverse operation if the value of the inverse operation
3 meets predetermined criteria.

1 14. The system as recited in claim 1, wherein included are six input buffers
2 coupled to the first input of the multiplication logic unit.

1 15. The system as recited in claim 1, wherein the multiplication logic unit
2 includes four multipliers coupled in parallel.

1 16. The system as recited in claim 1, wherein the arithmetic logic unit includes
2 three adders coupled in parallel and series.

1 17. The system as recited in claim 1, wherein the register unit includes four sets
2 of registers each having an output coupled to a first input of an associated
3 multiplexer which has a second input coupled to the input of the
4 corresponding set of registers.

1 18. The system as recited in claim 1, wherein the register unit is threaded.

1 19. The system as recited in claim 1, wherein the output converter is adapted to be
2 coupled to the lighting module via output buffers.

1 20. The system as recited in claim 1, wherein a register is coupled between the
2 output of the inverse logic unit and an input of the conversion unit.

1 21. The system as recited in claim 20, wherein the register is threaded.

1 22. The system as recited in claim 1, wherein the register unit is capable of being
2 masked at a vector component level.

1 23. A system for handling scalar and vector components during graphics
2 processing, comprising:
3 (a) a vector operation module for receiving vertex data in the form of vectors
4 and performing vector operations on the vector vertex data;

5 (b) a conversion module coupled to the vector operation module for converting
6 scalar vertex data from the vector operation module into vector vertex data;
7 and
8 (c) a register coupled to the vector operation module for storing an output of the
9 vector operation module for feeding the output back to the vector operation
10 module.

1 24. The system as recited in claim 23, wherein the vector operation module
2 includes at least one of multiplier and an adder.

1 25. The system as recited in claim 23, wherein zero latency is achieved by
2 bypassing the register.

1 26. The system as recited in claim 25, wherein the register includes a vector
2 component write mask for generating vector vertex data.

1 27. The system as recited in claim 23, and further comprising a scalar operation
2 module adapted for executing scalar operations on an output of the vector
3 operation module, thereby rendering vertex data in the form of scalars.

1 28. The system as recited in claim 27, wherein the scalar operations include
2 inverse or inverse square root operations.

1 29. A method for handling scalar and vector components during graphics
2 processing, comprising:
3 (a) receiving vertex data in the form of vectors;
4 (b) performing vector operations on the vector vertex data;
5 (c) converting scalar vertex data resulting from the vector operations into vector
6 vertex data;
7 (d) storing an output of the vector operations; and

8 (e) performing additional vector operations on the stored output of the vector
9 operations.

1 30. The method as recited in claim 29, wherein the vector operations include
2 multiplication or addition operations.

1 31. The method as recited in claim 29, wherein the vector operations are
2 performed on the output of the vector operations with zero latency.

1 32. The method as recited in claim 31, wherein the output of the vector
2 operations is stored in a register unit, and the zero latency is achieved by
3 bypassing the register unit.

33
1 34. The method as recited in claim 29, and further comprising executing scalar
2 operations on an output of the vector operations, thereby rendering vertex
3 data in the form of scalars.

34
1 35. The method as recited in claim 34, wherein the scalar operations include
2 inverse or inverse square root operations.

35
1 36. The method as recited in claim 34, and further comprising extracting scalar
2 vertex data from the output of the vector operations if the output is in the
3 form of vectors.

36
1 37. The method as recited in claim 36, wherein the extraction is carried out by a
2 multiplexer.

37
1 38. The method as recited in claim 29, wherein the received vertex data is
2 manipulated by a multiplexer.

38

1 39. A computer program embodied on a computer readable medium for handling
2 scalar and vector components during graphics processing, comprising:
3 (a) a code segment for receiving vertex data in the form of vectors;
4 (b) a code segment for performing vector operations on the vector vertex data;
5 (c) a code segment for converting scalar vertex data resulting from the vector
6 operations into vector vertex data;
7 (d) storing an output of the vector operations; and
8 (e) performing additional vector operations on the stored output of the vector
9 operations.

39

1 40. The computer program as recited in claim 39, wherein the vector operations
2 include multiplication or addition operations.

40

1 41. The computer program as recited in claim 39, wherein the vector operations
2 are performed on the output of the vector operations with zero latency.

41

1 42. The computer program as recited in claim 41, wherein the output of the
2 vector operations is stored in a register unit, and the zero latency is achieved
3 by bypassing the register unit.

42

1 43. The computer program as recited in claim 39, and further comprising a code
2 segment for executing scalar operations on an output of the vector
3 operations, thereby rendering vertex data in the form of scalars.

43

1 44. The computer program as recited in claim 43, wherein the scalar operations
2 include inverse or inverse square root operations.

44

1 45. The computer program as recited in claim 43, and further comprising a code
2 segment for extracting scalar vertex data from the output of the vector
3 operations if the output is in the form of vectors.

1 ~~75~~ **46** The method as recited in claim 45, wherein the extraction is carried out by a
2 multiplexer.

1 47. The method as recited in claim 39, wherein the received vertex data is
2 manipulated by a multiplexer.

47 48. A method for performing a blending operation during graphics processing in
1 a hardware-implemented graphics pipeline, comprising:
2
3 (a) receiving a plurality of matrices, a plurality of weight values each
4 corresponding with one of the matrices, and vertex data in a buffer;
5 (b) calculating a sum of a plurality of products with each product calculated by
6 the multiplication of the vertex data, one of the matrices, and the weight
7 corresponding to the matrix, wherein the calculation is executed on a single
8 integrated circuit; and
9 (c) outputting the sum of products for additional processing.

1 48. The method as recited in claim 48, wherein the matrices include model view
2 matrices.

49
50. The method as recited in claim 49, wherein the additional processing
includes multiplying the sum of products by a composite matrix for
displaying purposes.

50
1 51. The method as recited in claim 49, wherein the additional processing
2 includes a lighting operation.

1 51 The method as recited in claim 48, wherein the matrices include inverse
2 52 matrices and the vertex data includes a normal vector.

52
1 53. The method as recited in claim 48, wherein the single integrated circuit
2 includes: a multiplication logic unit having a first input coupled to an output

3 of the buffer for receiving the vertex data; an arithmetic logic unit having a
4 first input coupled to an output of the multiplication logic unit; a register unit
5 having an input coupled to an output of the arithmetic logic unit; memory
6 coupled to the multiplication logic unit and the arithmetic logic unit, the
7 memory having stored therein a plurality of constants and variables for being
8 when processing the vertex data.

1 ~~53~~ ⁵⁴ The system as recited in claim 48, wherein the single integrated circuit
2 includes: a multiplication logic unit having a first input coupled to an output
3 of the buffer; an arithmetic logic unit having a first input coupled to an
4 output of the multiplication logic unit; a register unit having an input coupled
5 to an output of the arithmetic logic unit; an inverse logic unit including an
6 input coupled to the output of the arithmetic logic unit or the register unit for
7 performing an inverse or an inverse square root operation; a conversion
8 module coupled between an output of the inverse logic unit and a second
9 input of the multiplication logic unit, the conversion module adapted to
10 convert scalar vertex data to vector vertex data; memory coupled the
11 multiplication logic unit and the arithmetic logic unit, the memory having
12 stored therein a plurality of constants and variables for being used when
13 processing the vertex data; and an output converter coupled to the output of
14 the arithmetic logic unit for being coupled to a lighting module to output the
15 processed vertex data thereto.

1 ~~55~~ ⁵⁴ A system for performing a blending operation during graphics processing in a
2 graphics pipeline, comprising:
3 (a) a buffer for receiving a plurality of matrices, a plurality of weight values each
4 corresponding with one of the matrices, and vertex data;
5 (b) a single integrated circuit coupled to the buffer for calculating a sum of a
6 plurality of products with each product calculated by the multiplication of the
7 vertex data, one of the matrices, and the weight corresponding to the matrix;
8 and

9 (c) wherein the sum of products is outputted from the single integrated circuit
10 for additional processing.

1 ~~55~~ ⁵⁶ The system as recited in claim 55, wherein the matrices include model view
2 matrices.

1 ~~57~~ ⁵⁸ The system as recited in claim 56, wherein the additional processing includes
2 multiplying the sum of products by a composite matrix for displaying
3 purposes.

1 ~~58~~ ⁵⁷ The system as recited in claim 56, wherein the additional processing includes
2 a lighting operation.

1 ~~59~~ ⁵⁸ The system as recited in claim 55, wherein the matrices include inverse
2 matrices and the vertex data includes a normal vector.

1 ~~60~~ ⁵⁹ The system as recited in claim 55, wherein the single integrated circuit
2 includes: a multiplication logic unit having a first input coupled to an output
3 of the buffer for receiving the vertex data; an arithmetic logic unit having a
4 first input coupled to an output of the multiplication logic unit; a register unit
5 having an input coupled to an output of the arithmetic logic unit; memory
6 coupled to the multiplication logic unit and the arithmetic logic unit, the
7 memory having stored therein a plurality of constants and variables for being
8 used when processing the vertex data.

1 ~~61~~ ⁶⁰ The system as recited in claim 55, wherein the single integrated circuit
2 includes: a multiplication logic unit having a first input coupled to an output
3 of the buffer; an arithmetic logic unit having a first input coupled to an
4 output of the multiplication logic unit; a register unit having an input coupled
5 to an output of the arithmetic logic unit; an inverse logic unit including an
6 input coupled to the output of the arithmetic logic unit or the register unit for

7 performing an inverse or an inverse square root operation; a conversion
8 module coupled between an output of the inverse logic unit and a second
9 input of the multiplication logic unit, the conversion module adapted to
10 convert scalar vertex data to vector vertex data; memory coupled to the
11 multiplication logic unit and the arithmetic logic unit, the memory having
12 stored therein a plurality of constants and variables for being used when
13 processing the vertex data; and an output converter coupled to the output of
14 the arithmetic logic unit for being coupled to a lighting module to output the
15 processed vertex data thereto.

Subj 1

1 ~~61~~ ~~62~~ A method for handling output values in a graphics processing module
2 representative of an inverse operation involving a W-attribute of vertex data,
3 comprising:
4 (a) processing vertex data, wherein the processing of the vertex data includes an
5 inverse operation involving a W-attribute of the vertex data;
6 (b) outputting the processed vertex data;
7 (c) identifying a value of the ~~inverse~~ operation involving the W-attribute of the
8 vertex data; and
9 (d) clamping the value of the inverse operation if the value of the inverse
10 operation meets predetermined criteria.

1 ~~62~~ ~~63~~ The method as recited in claim 62, wherein the criteria includes the value of
2 the inverse operation being greater than a predetermined amount.

1 ~~63~~ ~~64~~ The method as recited in claim 62, wherein the value is clamped by an
2 inverse logic unit in a transform module.

1 ~~64~~ ~~65~~ The method as recited in claim 62, wherein the value is clamped to a
2 minimum and a maximum exponent.

Sub 65

1 65, A computer program embodied on a computer readable medium for handling
2 output values in a graphics processing module representative of an inverse
3 operation involving a W-attribute of vertex data, comprising:
4 (a) a code segment for processing vertex data, wherein the processing of the
5 vertex data includes an inverse operation involving a W-attribute of the
6 vertex data;
7 (b) a code segment for outputting the processed vertex data;
8 (c) a code segment for identifying a value of the inverse operation involving the
9 W-attribute of the vertex data; and
10 (d) a code segment for clamping the value of the inverse operation if the value of
11 the inverse operation meets predetermined criteria.

66

1 66, The computer program as recited in claim 66, wherein the criteria includes
2 the value of the inverse operation being greater than a predetermined amount.

67

1 67, The computer program as recited in claim 66, wherein the value is clamped
2 by an inverse logic unit in a transform module.

68

1 68, The computer program as recited in claim 66, wherein the value is clamped
2 to a minimum and a maximum exponent.